Project 2

S.O.S. Morse Code Signaler

Project 2 - SOS Morse Code Signaler

What you will need

For this project we are going to leave the exact same circuit set up as in Project 1, but will use some different code to make the LED display a message in Morse Code. In this case, we are going to get the LED to signal the letters S.O.S., which is the international morse code distress signal. Morse Code is a type of character encoding that transmits letters and numbers using patterns of On and Off. It is therefore nicely suited to our digital system as we can turn an LED on and off in the necessary pattern to spell out a word or a series of characters. In this case we will be signaling S.O.S. which in the Morse Code alphabet is three dits (short flash), followed by three dahs (long flash), followed by three dits again.

We can therefore now code our sketch to flash the LED on and off in this pattern, signaling SOS.

```
// Project 2 - SOS Morse Code Signaler

// LED connected to digital pin 10
int ledPin = 10;

// run once, when the sketch starts
void setup()
{
   // sets the digital pin as output
   pinMode(ledPin, OUTPUT);
}

// run over and over again
void loop()
{
  // 3 dits
  for (int x=0; x<3; x++) {
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(150);                  // waits for 150ms
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(100);                // waits for 100ms
  }

  // 100ms delay to cause slight gap between letters
  delay(100);
  // 3 dahs
  for (int x=0; x<3; x++) {
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(400);                  // waits for 400ms
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(100);                // waits for 100ms
  }

  // 100ms delay to cause slight gap between letters
  delay(100);

  // 3 dits again
  for (int x=0; x<3; x++) {
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(150);                  // waits for 150ms
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(100);                // waits for 100ms
  }

  // wait 5 seconds before repeating the SOS signal
  delay(5000);
}
```

Enter the code

Create a new sketch and then type in the code listed above. Verify your code is error free and then upload it to your Arduino.

If all goes well you will now see the LED flash the Morse Code SOS signal, wait 5 seconds, then repeat.

If you were to rig up a battery operated Arduino to a very bright light and then place the whole assembly into a waterproof and handheld box, this code could be used to control an SOS emergency strobe light to be used on boats, whilst mountain climbing, etc.

So, let's take a look at this code and work out how it works.

Project 2 - Code Overview

So the first part of the code is identical to the last project where we initialise a variable and then set pin 10 to be an output. In the main code loop we can see the same kind of statements to turn the LED's on and off for a set period of time, but this time the statements are within 3 separate code blocks.

The first block is what outputs the 3 dits

```
for (int x=0; x<3; x++) {
digitalWrite(ledPin, HIGH);
delay(150);
digitalWrite(ledPin, LOW);
delay(100);
}
```

We can see that the LED is turned on for 150ms and then off for 100ms and we can see that those statements are within a set of curly braces and are therefore in a separate code block. But, when we run the sketch we can see the light flashes 3 times not just once.

This is done using the for loop.

```
for (int x=0; x<3; x++) {
```

This statement is what makes the code within it's code block execute 3 times. There are 3 parameters we need to give to the for loop. These are initialisation, condition, increment. The **initialisation** happens first and exactly once. Each time through the loop, the **condition** is tested; if it's true, the statement block, and the **increment** is executed, then the **condition** is tested again. When the **condition** becomes false, the loop ends.

So, first we need to initialise a variable to be the start number of the loop. In this case we set up variable X and set it to zero.

```
int x=0;
```

We then set a condition to decide how many times the code in the loop will execute.

```
x<3;
```

In this case the code will loop if X is smaller than (<) 3. The code within a for loop will always execute once no matter what the condition is set to.

The < symbol is what is known as a 'comparison operator'. They are used to make decisions within your code and to compare two values. The symbols used are:-

- == (equal to)
- != (not equal to)
- < (less than)
- > (greater than)
- <= (less than or equal to)
- >= (greater than or equal to)

In our code we are comparing x with the value of 3 to see if it is smaller than 3. If x is smaller than 3, then the code in the block will repeat again.

The final statement is

```
x++
```

this is a statement to increase the value of x by 1. We could also have typed in `x = x + 1;` which would assign to x the value of x + 1. Note there is no need to put a semi-colon after this final statement in the for loop.

You can do simple mathematics using the symbols +, -, * and / (addition, subtraction, multiplication and division). E.g.

```
1 + 1 = 2
3 - 2 = 1
2 * 4 = 8
8 / 2 = 4
```

So, our for loop initialises the value of x to 0, then runs the code within the block (curly braces). It then increases the increment, in this case adds 1 to x. Finally it then checks that the condition is met, which is that x is smaller than 3 and if so repeats.

So, now we know how the for loop works, we can see in our code that there are 3 for loops, one that loops 3 times and displays the 'dits', the next one repeats 3 times and displays the 'dahs', then there is a repeat of the dit's again.

It must be noted that the variable x has a local 'scope', which means it can only be seen by the code within it's own code block. Unless you initialise it before the setup() function in which case it has 'global scope' and can be seen by the entire program. If you try to access x outside the for loop you will get an error.

In between each for loop there is a small delay to make a tiny visible pause between letters of SOS. Finally, the code waits for 5 seconds before the main program loop starts again from the beginning.

OK now let's move onto using multiple LED's.